# project 2

Piere 2022-11-21

Exercise 1:Marketing, conversions and conversions lags

The data contains who buys on that site (conversions). The data has been simplified a lot, but you can imagine it being from a site that rents out rooms for hotels. The database represents customers coming from a search engine (e.g. google), pressing an ad (a “click”) and whether or not they bought something (if a conversion occurred). There are two tables, “Clicks” and “Conversion_value”. Each id represent a unique customer. Our following tasks will be:

a)Visualize the number of clicks made on each day (date).

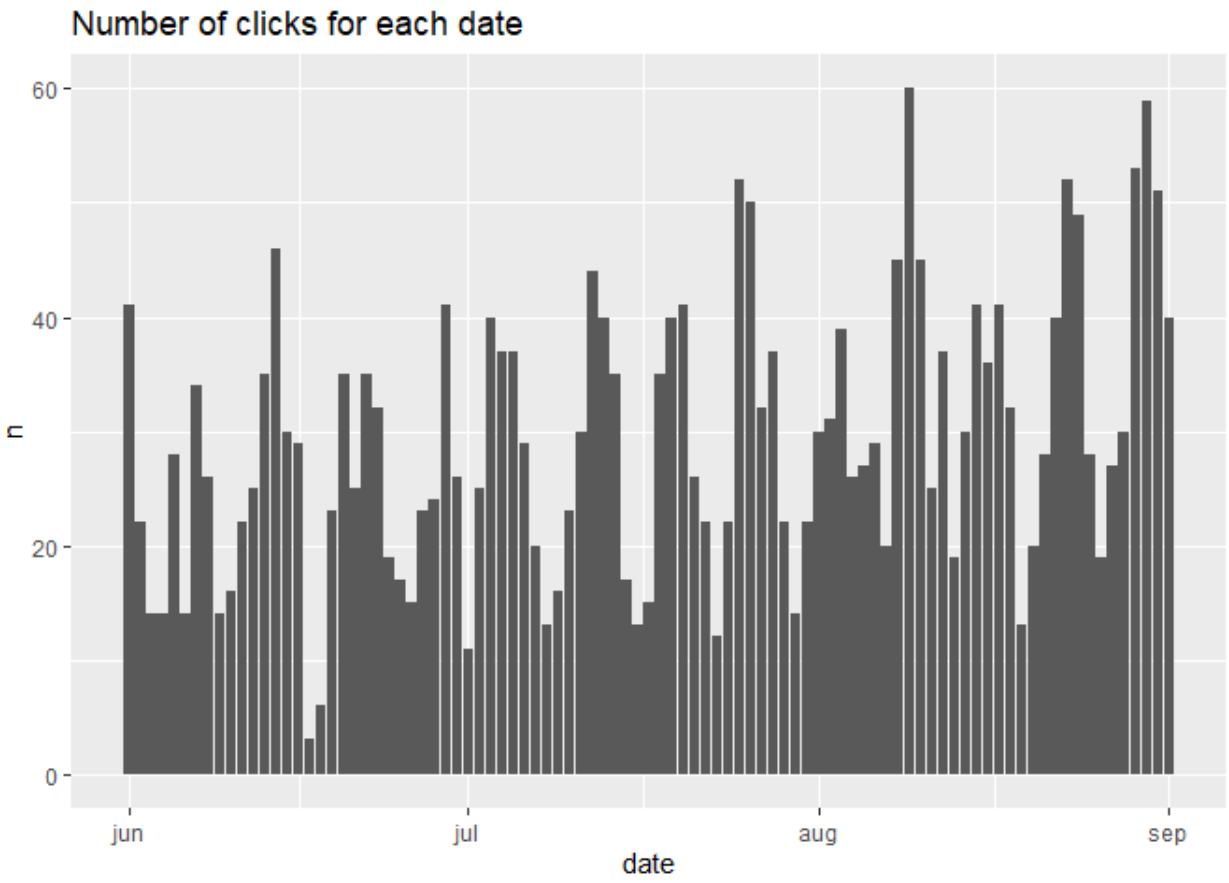
b)Given that there is a cost to each click, which day of the week has been most costly? Hint: there are many ways to solve to problem, but have a look at lubridate::wday.

c)Make a histogram of the conversion value customers provided.

d)The “Conversion lag” (CL) of a customer is the number of days from the customer clicked on an ad to the customer bought something/converted. An example is that the customer clicked an add on the first on January and rented a room on the third of January. The CL is then equal to 2. Plot its distribution in R, interpret and comment on the results. Hint: Google on how to take differences of dates. If you want to write a full SQL solution you can use julianday.

e)Present two tables on the following format

1a)



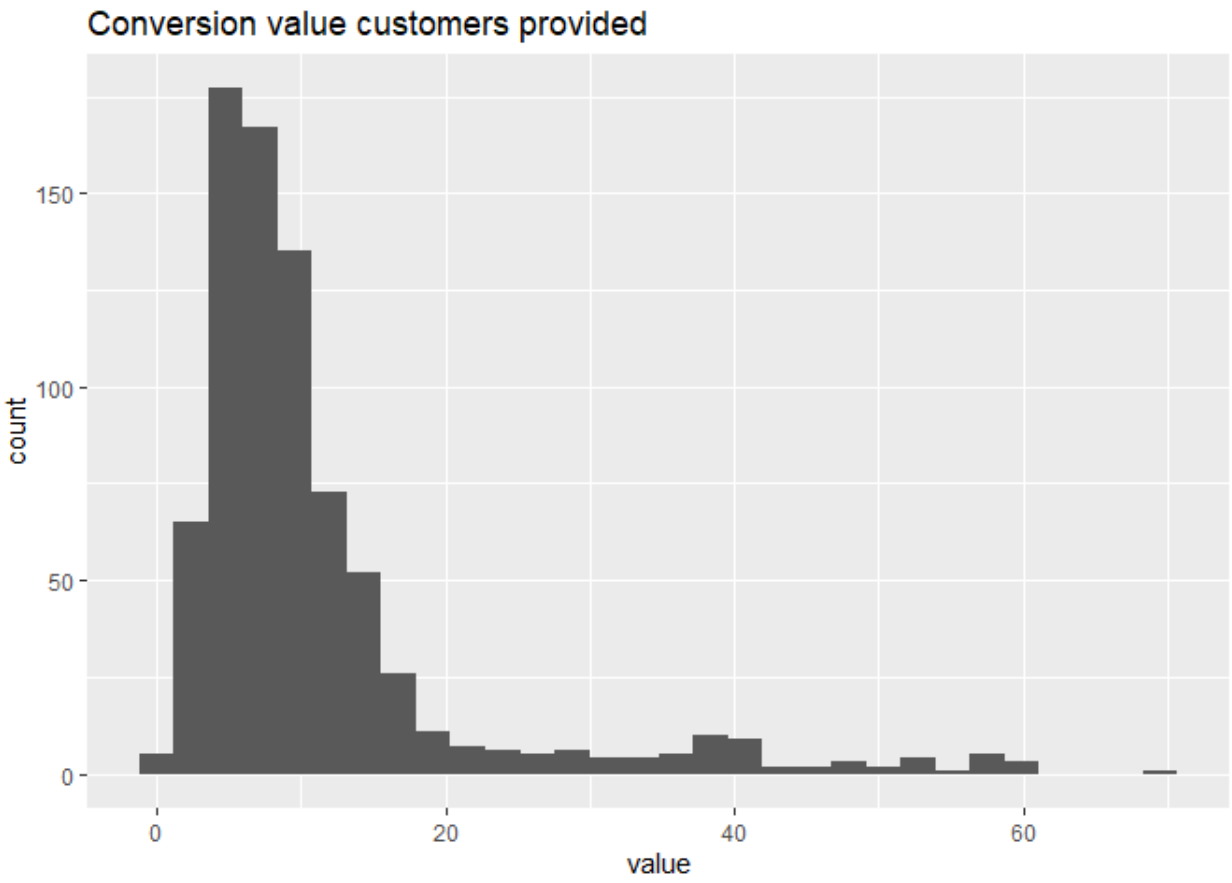

A visualization of the number of clicks for each date.

1b)

| date | n |
|---|---|
| sön | 538 |
| mån | 540 |
| tis | 392 |
| ons | 233 |
| tor | 234 |
| fre | 321 |
| lör | 470 |

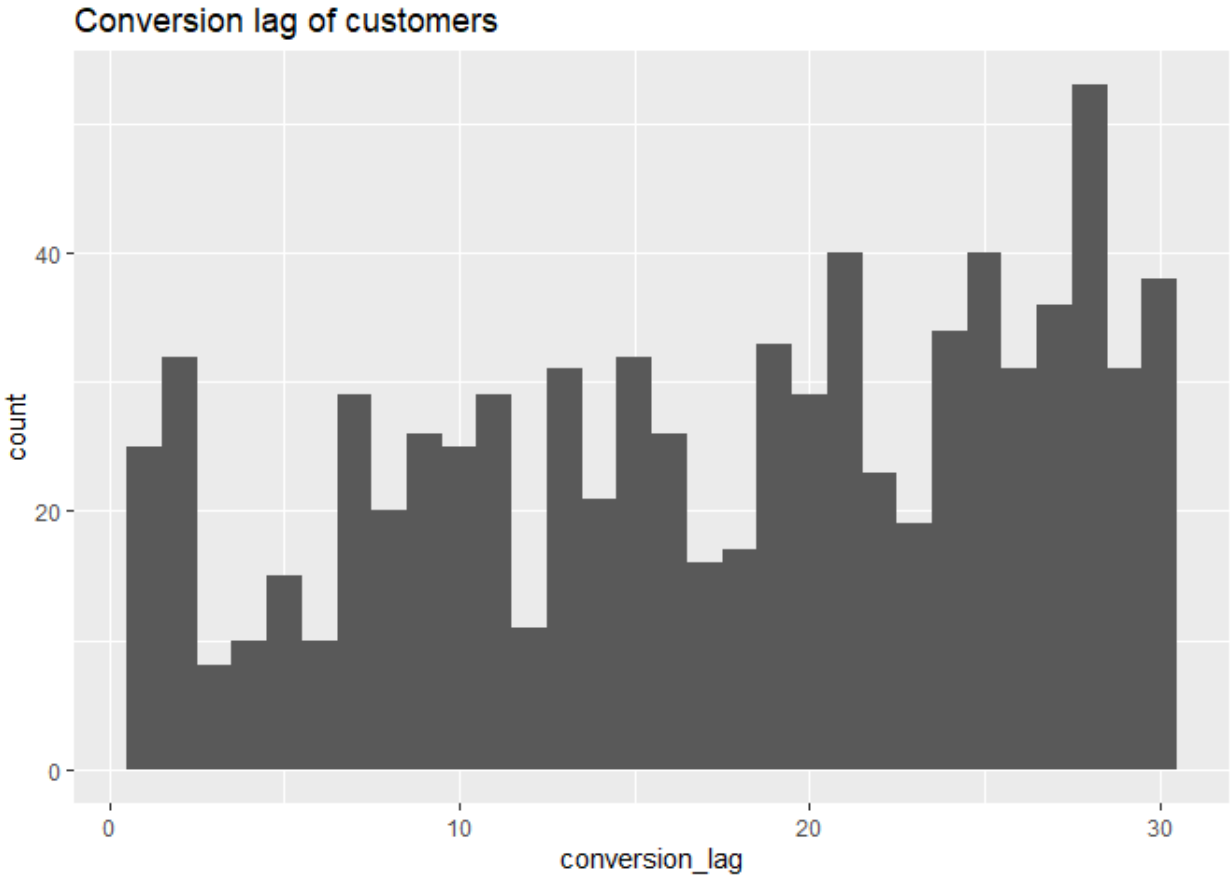
We can see in the table that monday has been the most costly day.

1c)




A visualization of the conversion value for each customer that provided.

1d)



A visualization over the “conversion lag” of a customer.

1e)

| date.clicked | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| 2020-08-28 | 0 | 2.3894214 | 0.0000000 | 0 | 0.0000000 | 0.000000 | 0.0 |
| 2020-08-29 | 0 | 0.0000000 | 0.0000000 | 0 | 0.0000000 | 0.000000 | 0.4 |
| 2020-08-30 | 0 | 0.5351251 | 0.0000000 | 0 | 0.7031652 | 0.000000 | 2.3 |
| 2020-08-31 | 0 | 0.0000000 | 0.0000000 | 0 | 0.0000000 | 1.228696 | 0.0 |
| 2020-09-01 | 0 | 0.6367069 | 0.9622497 | 0 | 0.0000000 | 0.000000 | 0.0 |

| date.clicked | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2020-08-28 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 2020-08-29 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 2020-08-30 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 2 | 0 | 2 | 0 | 0 | 0 |
| 2020-08-31 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 2020-09-01 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |

The first table shows the average of the conversion value for each CL during a certain date. The second table shows the sum of number of clicks for each CL during a certain date.

Exercise 2: SL lines

Our data contains all SL’s current stops, lines, and the stops on each line. The data are obtained from a call to TrafikLab’s SL Hållplatser och Linjer 2 API on 2019-11-17 using the httr package. Note: Trafiklab has further documentation of the API and the variables.Our tasks are the following:

a)b) List all variables and figure out and describe how the tables relate to each other. c) Present a table of the number of active unique rail traffic stops (i.e. train, tram or metro stops in each ticket zone (ZoneShortName in stopAreas/stopPoints). By “active” we mean stops that are part of the journey pattern (as defined in journeyPatterns) of a line. d) Choose a line, and plot the stops as points on a map with the name of each stop as a label. Write the code in such a way that it is easily reusable if you want to plot another

line. In order to produce a map use the latitude and longitude coordinates and generate the plot using leaflet package. e)Consider the stopAreas and stopPoints tables and comment on the sparsity of this data presentation, e.g., are there any (unnecessary) redundancies? Suggest a more sparse data model for the stopAreas table and perform the appropriate table operations to obtain this sparser representation and store it in the data.frame stopAreas_sparse. Explain how one would get the original stopAreas data.frame using joins. Note: A very operational way to check memory consumption for storing a variable in R is to use the object.size function.

2a,b)

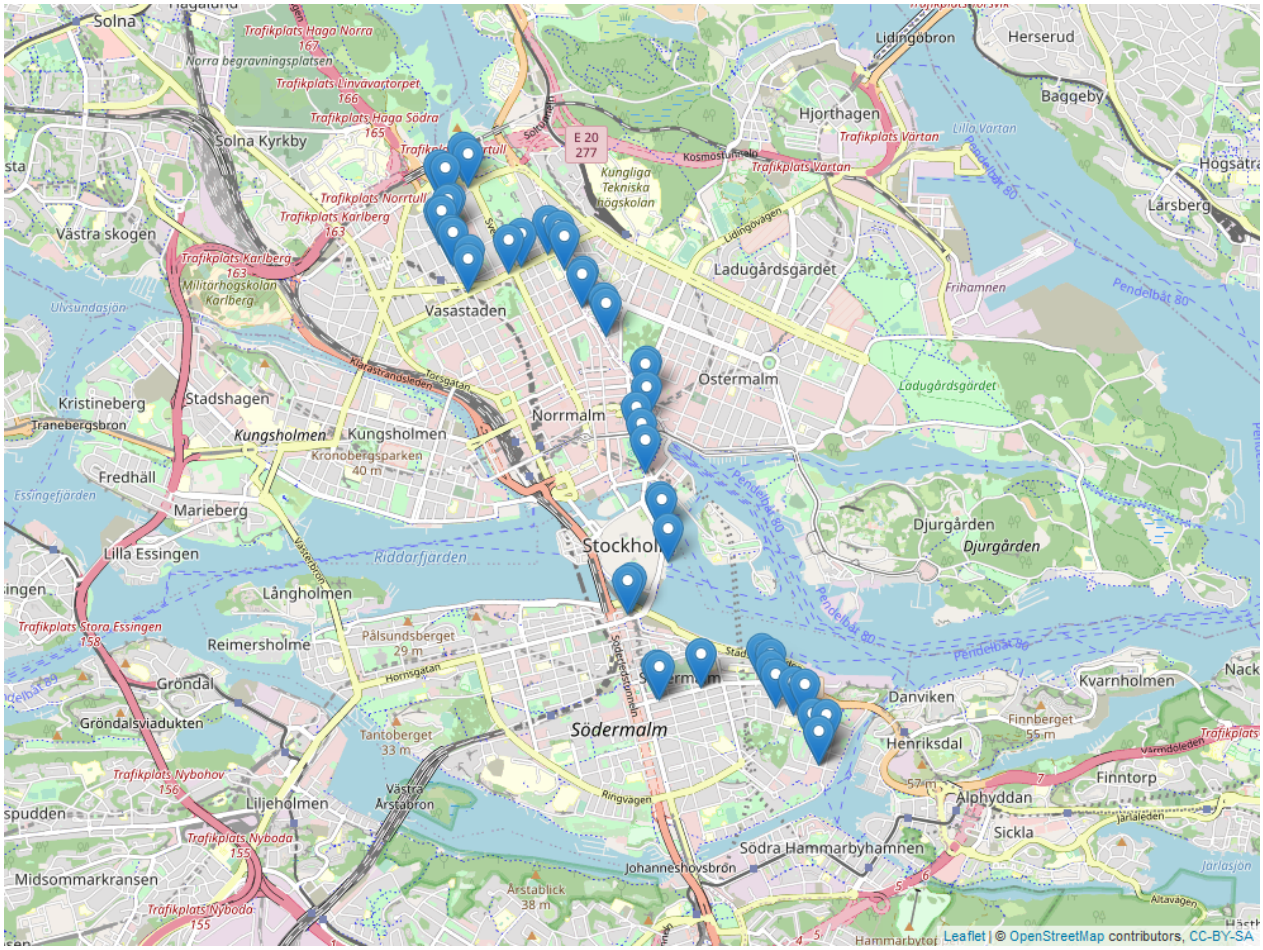
JourneyPatterns describes a unique journey pattern for a line number. Lines describe an overview of which line number it is, also what kind of transportation for example bus or train. StopAreas describes the names of the stop checkpoints for the transportation also in which zone you are located. Sites pretty summarizes pretty much stopAreas. These are grouped and described as site names. Stoppoints shows where the transportation will make its stop in a certain zone. The coordinates of the stop location is also give as northing and easting coordinates. Finally transportmodes describes the mode of transportation either bus, boat, metro or tram.

2c)

| ZoneShortName | METROSTN | RAILWSTN | TRAMSTN |
|---|---|---|---|
| A | 232 | 45 | 169 |
| B | 0 | 60 | 69 |
| C | 0 | 40 | 0 |

A summary of the rail traffic stops in each ticket zone.

2d)




A visualization of line 2, Barnängen (starting from the bottom), all the way to Sveaplan (ending at the top) and its stop points.

2e)

Both tables are identical to each other. I did an object size on both tibbles and they had the same amount of bytes. I also used a function called all.equal which tells us if the two tibbles are identical which came to be true. However thought the stop areas has doublets if not more of the same stop points and these are seen as redundant. It perhaps has to do with the stop stations and could be that there are several stations in the same area with the same name. Joining by the stopAreaNumber with help of a inner join we can then get rid of the stop point number because we already know the stop checkpoint it is made by the stop area number.